Predicting Transient Response using Data-Driven Models for Ball-Impact Simulations

Ross Pivovar
Advanced Computing
AWS
12 W 39th St, New York, NY 10018
rpivovar@amazon.com

Fei Chen
Advanced Computing
AWS
3075 Olcott St, Santa Clara, CA 95054
fchenaws@amazon.com

Raghunath Katragadda
Product Integrity
Lab126
1100 Enterprise Way, Sunnyvale, CA 94089
katragar@lab126.com

Vidyasagar Ananthan *
Advanced Computing
AWS
2205 7th Ave, Seattle, WA 98121
vidyaaws@amazon.com

Abstract

This study investigates the application of machine learning (ML) models for predicting transient responses in ball-impact elastodynamics simulations. We focus on the canonical problem of ball impact on laminated structures, which captures essential physics while maintaining computational tractability. Novel contributions include: (1) development of a temporal multi-resolution strategy for stable long-time predictions, (2) systematic comparison of U-Nets and Fourier Neural Operators as spatial ML kernels, and (3) demonstration of accurate non-local metric predictions across full time-horizons. Using a synthetic dataset of 6500 impact scenarios, we achieve 3.5-8% prediction accuracy while providing 10,000x speedup compared to traditional FEM simulations. The proposed methodology enables rapid virtual prototyping for impact-resistant design optimization.

1 Introduction

Accurately characterizing and predicting the mechanical response of heterogeneous structures under dynamic and impact loading conditions is crucial across industries from consumer electronics [1–3], to automotive [4] and aerospace [5] sectors. In consumer electronics, devices undergo rigorous ball-impact and edge-drop testing to evaluate structural performance, compliance and impact resistance, simulating real-world scenarios during daily usage [6]. Modeling this behavior is challenging due to complexities like strain-rate effects, material nonlinearities, wave propagation, interfacial contact mechanics and deformation localization [7–9].

Traditionally, numerical techniques for solving the governing elastodynamics equations, such as incremental Finite Element Methods (FEM) [10], have been used in tandem with material characterization experiments to model the mechanics of impact and dynamic loading. These FEM simulations are implemented with either explicit time-marching schemes or implicit methods [11, 12], where the former require bounds on admissible time-discretization levels to ensure numerical stability [13] while minimizing the accumulation of error through the time-marching process [14] and capturing the critical time-scales of the physical phenomena. Consequently, these simulations are computationally

* Author to whom any correspondence should be addressed.

expensive for predicting longer-timescale phenomena during dynamic impact loading. This work adopts a data-driven approach towards predicting the dynamic response of structures under impact loading, and compares the speed, accuracy and stability of various methods.

Recent advances in machine learning (ML) have accelerated the development of surrogate models for physical systems governed by partial differential equations (PDEs). Raissi et al. [15] first introduced Physics-Informed Neural Networks (PINNs), which directly embed physical laws as constraints into neural network loss functions, allowing networks to learn solutions while conforming to conservation principles and approximately satisfying boundary conditions.

In order to overcome scaling limitations with pure physics-informed methods, data-driven approaches have been investigated which learn from past PDE solutions without explicit information about the governing equations. Ronneberger et al. [16] developed the CNN-based U-Net architecture, initially for biomedical image segmentation and other computer vision tasks, further extended by Çiçek et al. [17] for 3D applications. Neural operators, a more advanced data-driven approach in which networks learn on continuous function spaces, were independently developed by Lu et al. [18, 19] in the form of DeepONet, and by Li et al. [20] as Fourier Neural Operators (FNOs).

As a hybrid approach bridging physics-informed and data-driven methods, Li et al. [21] developed Physics-Informed Neural Operators (PINO), which combined the spectral efficiency of FNOs with physical constraints for more stable and accurate predictions. Li et al. [22] later extended this framework to generalized geometries in irregular spaces mapped to uniform grids in latent space, creating geo-FNOs, which were then extended to Geometry-Informed Neural Operators (GINOs) by combining FNOs with graph neural operators to achieve discretization-convergent solutions [23].

In terms of applications, prior work has been done by Rao et al. [24] who have used PINNs for computational elastodynamics with a mixed-variable (displacement and stress) formulation and imposition of initial and boundary conditions as hard constraints through a composite scheme of deep neural networks. Rasht-Behesht et al. [25] have used PINNs for wave propagation and waveform inversion for seismic applications. In a data-driven approach, Lehmann et al. [26] have shown that factorized FNOs (f-FNOs) can be used model surface elastic wave propagation in 3D spatiotemporal domains.

The accumulation and propagation of error through sequential time-slice predictions, particularly for long time-horizon problems, is a critical challenge when modeling transient mechanics. Kochkov et al. [27] have demonstrated that incorporating elements of physics into an ML approach improves temporal stability and spatial accuracy for fluid mechanics predictions. From a data-driven perspective Lippe et al. [28, 29] have introduced a novel approach called PDE-Refiner to address this challenge by using a multi-step refinement process inspired by diffusion models. Considering autoregressive time sequence prediction of spatial variables, Takamoto et al. [30] devised a pushforward training strategy where gradients only flow through the last timestep and show that this stabilizes U-Net autoregressive predictions for longer time-horizons. Ghule et al. [31] show the effectiveness of Curriculum Learning, inspired by Natural Language Processing (NLP) training methods [32] in the context of predicting transient simulations.

In this study, we aim to understand how different underlying spatial machine learning surrogate models and temporal training strategies incur accumulated error for the transient elastodynamics problem of ball-impact on a heterogeneous laminate structure. The focus of this work, and its inherent novelty, lies in three key aspects:

1. The first systematic investigation of numerical stability for long-horizon transient predictions using machine learning (ML) models for non-linear finite-strain elastodynamics problems
2. The development of a novel temporal multi-resolution strategy to maintain stability during long-time predictions
3. A comprehensive comparison of spatial ML kernels (U-Nets vs FNOs) as base models in a temporal sequence prediction for elastodynamics applications

These advances address fundamental challenges in applying ML to industrial ball-impact structural simulations. To accomplish this, we (a) generate a synthetic ground truth dataset of 6500 samples across different geometric variations of plate inclusion using the open-source FEM solver MOOSE [33–35], (b) contrast various machine learning architectures including U-Nets and FNOs (c) investigate the inclusion of global parameters such as kinetic energy into the ML models, (d) explore

training strategies including recurrence and temporal multi-resolution and finally (e) understand generalization error on unseen geometries.

2 Problem Formulation

We consider an elastic body contained within a volume $\Omega \in \mathbb{R}^3$ (shown in Figure 1) subject to a dynamic impact load over a specified time period $t \in [0, t_N]$, where N specifies a number of uniform discretization steps in time. The diameter of the ball is chosen $D \gg l$ where l is the characteristic length scale (i.e. thickness) of the body to ensure reasonable discretization insensitivity.

The deformation of any point $\mathbf{X} \subset \Omega$ contained within the body in the undeformed configuration is described by the deformation map $\varphi(\mathbf{X}, t) : \Omega \times t \rightarrow \mathbb{R}^3$. Defining the displacements $\mathbf{u} = \varphi(\mathbf{X}, t) - \mathbf{X}$, material time-derivative velocity $\mathbf{v} = \dot{\mathbf{u}}$, Cauchy stress tensor σ, density ρ, and gravitational force $\mathbf{g}$, conservation of linear momentum requires that

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \ \sigma + \rho \mathbf{g} \tag{1}$$

We utilize a Zaremba-Jaumann objective stress rate [36] formulation to compute the Cauchy stress [37],

$$\sigma^{\nabla J} = \frac{D\sigma}{Dt} - \mathbf{W} \cdot \sigma - \sigma \cdot \mathbf{W}^T, \tag{2}$$

$$\mathbf{W} = \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^T\right), \qquad \mathbf{L} = (\nabla \mathbf{v})^T \tag{3}$$

The constitutive relationship is assumed to be isotropic elastic, relating the rate-of-deformation tensor $\mathbf{D}$ to the stress measure $\sigma^{\nabla J}$

$$\sigma^{\nabla J} = \mathbb{C} : \mathbf{D} \tag{4}$$

The fourth-order elasticity tensor $\mathbb{C}$ can be expressed in terms of material properties Young's Modulus E and Poisson's ratio ν, where

$$\mathbb{C}_{ijkl} = \frac{E\nu}{(1+\nu)(1-2\nu)}\delta_{ij}\delta_{kl} + \frac{E}{2(1+\nu)}\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \tag{5}$$

$$\delta_{nm} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} e^{2\pi i \frac{k}{N}(n-m)} \tag{6}$$

In order to compute the rate-of-deformation tensor, we begin with the incremental deformation tensor, with the deformation gradient at a given time t_n, given by $\mathbf{F}_n = \nabla_{\mathbf{X}} \varphi(X, t_n)$ and incremental right Cauchy-Green deformation tensor $\hat{\mathbf{C}}$ [38],

$$\hat{\mathbf{F}} = \mathbf{F}_n \left(\mathbf{F}_{n-1}\right)^{-1} \tag{7}$$

$$\hat{\mathbf{C}} = \hat{\mathbf{F}}^T \hat{\mathbf{F}} \tag{8}$$

$$\mathbf{D} = \frac{1}{\Delta t} \log(\hat{\mathbf{C}}^{1/2}) \tag{9}$$

A Taylor series expansion is used to simplify and approximate the calculation of $\mathbf{D}$

$$\mathbf{D} = \frac{1}{\Delta t}\left(-\frac{1}{2}(\hat{\mathbf{C}}^{-1} - \mathbb{I}) + \frac{1}{4}(\hat{\mathbf{C}}^{-1} - \mathbb{I})^2 - \frac{1}{6}(\hat{\mathbf{C}}^{-1} - \mathbb{I})^3 + ...\right) \tag{10}$$

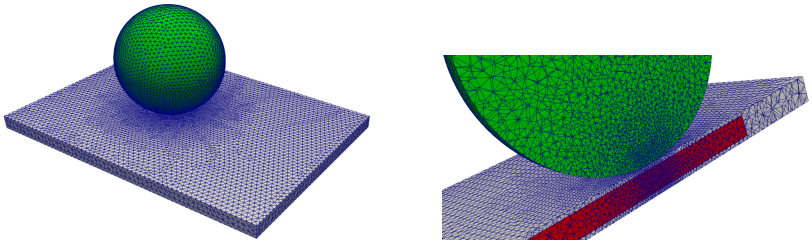

Figure 1: Ball impact scenario: the experimental setup showing relative sizes of the ball and plate (left) and the embedded inclusion within the plate (right). The mesh overlay conveys element sizes at various points within the mesh.

In order to generate synthetic training data, we run numerical simulations on a solid plate with an embedded laminate inclusion impacted by a spherical ball. As the first step towards building an spatio-temporal surrogate ML model to accurately approximate the simulation results, we focus on designs parameterized by 8 variables $\Theta = [X_b, Y_b, X_i, Y_i, Z_i, L_i, D_i, H_i]$: the ball location (X_b, Y_b), inclusion centroid location (X_i, Y_i, Z_i), and inclusion dimensions (L_i, D_i, H_i). Further details on generating synthetic data used for training ML models are provided in Appendix A.1.

In terms of developing the machine learning model, a similar approach is followed in terms of predicting incremental displacement and strain fields at each given time-step.

During training time, the ML model reduces the mean-squared errors between the simulation result and its prediction at a given time increment. The ML model takes as inputs a set of design parameters, $\Theta \in \Theta_{\text{train}}$, at a given timestep t_n, based on the state of the system variables (displacements, material properties and geometry) at the previous time t_{n-1} (or $t_{n-k},\ k > 1$ when exploring the influence of larger timestep sizes in numerical stability and accuracy).

At inference time, the ML model makes prediction on a given new design parameterized $\Theta \in \Theta_{\text{test}}$, and the starting conditions t_{n-1} (or t_{n-k}) in a time segment contained within $t_n \in [0, t_N]$, reminiscent of an initial-boundary value problem (IVBP) from the perspective of classical numerical analysis.

Extension of the ideas presented here to additional complexity (non-linear material properties, and internal geometric topology) is considered beyond the scope of this paper, and will be explored as part of future work.

3 Challenges

There are a number of challenges associated with developing spatio-temporal deep learning models for transient structural dynamics, associated with practical considerations from a computational perspective. These include:

1. **Spatial Complexity:** Developing a surrogate ML model for continuum simulations that can handle non-local interactions, multi-mesh assemblies, contact mechanics, large deformations, and discontinuous material properties, while ensuring mechanical compatibility, stress equilibrium, and computational efficiency.
2. **Accuracy in Transient Prediction:** Capturing the temporal dynamics of elastic wave propagation, including reflection, refraction and scattering phenomena within the timestep in terms of non-local error metrics across the spatial domain. Generally, FNO approaches are known to outperform convolutional neural networks (CNN)-based U-Nets for these single-step or short horizon roll-out problems [30].
3. **Numerical Stability:** Ensuring that the ML methodology remains numerically viable across long time-horizons, and does not amplify certain error spectra to result in temporal numerical instabilities. Neural Operators, in particular, especially FNOs with truncated spectra, are known to be unstable for long autoregressive rollouts [39].

In addition to these challenges, there is bias in training datasets created based on industrial designs in engineering practice, for instance in consumer electronics testing. This is due to a non-uniform

sampling of parametric design spaces due to empirical engineering intuition or aesthetic considerations. Whilst beyond the scope of this work, an *a priori* methodology of quantifying and avoiding bias in training data for scientific machine learning models would be highly beneficial for mechanics. Additionally, this would allow for the generation of synthetic data to augment regions which are data-deficient, and help with training improved machine learning models.

4 Machine Learning Methodology

In contrast to equilibrium or steady-state simulation problems, for which spatial machine learning kernels are sufficient for surrogate modeling, there is an additional component of time-marching and computing transient solutions. In this section we explore both of these aspects, as well as embedding non-local information about the state of the elastodynamic system (i.e. kinetic energy).

4.1 Spatial Approximation Kernels

Two spatial machine learning kernels are used in this comparison: FNOs and CNN-based U-Nets. This section briefly expands on the underlying architecture of each kernel for a single time-step rollout, and the subsequent section highlights the time-marching strategies.

4.1.1 Fourier Neural Operators

Fourier Neural Operators (FNOs) take advantage of the equivalence between difference operations, which can be expressed as convolutions, in the spatial domain and point-wise multiplication in the reciprocal frequency (Fourier) domain [20, 22]. The input data $u_t(x)$, representing the structural response (i.e. displacement components) at time t and coordinate $\mathbf{X}$, is transformed using a lifting layer feed-forward neural network P:

$$v_{t_n}(\mathbf{X}) = P(u_{t_n}(\mathbf{X})) \tag{11}$$

The input $v(\mathbf{X})$ is transformed to the reciprocal frequency domain using the Fourier transform, which can be calculated efficiently using Fast-Fourier Transform (FFT) algorithm

$$\hat{v}_{t_n}(\xi) = \mathcal{F}[v_{t_n}](\xi) \tag{12}$$

where ξ represents the frequency space. Typically, in order to improve training convergence, higher frequencies in ξ are truncated, retaining only the lower frequencies. An inverse transform maps back to the spatial domain, after applying a linear operator R:

$$\left[v_{t_n}(\mathbf{X})\right]_F = \mathcal{F}^{-1}\left[R\hat{v}_{t_n}(\xi)\right](\mathbf{X}) \tag{13}$$

As with typical deep neural networks, a bias term W and a non-linear activation function σ (e.g., ReLU or Softmax) are applied in the solution domain:

$$v_{t_{n+1}}(\mathbf{X}) = \sigma\left(W \cdot v_{t_n}(\mathbf{X}) + \left[v_{t_n}(\mathbf{X})\right]_F\right) = \sigma \circ K(v_{t_n}) \tag{14}$$

Here, K denotes the Fourier layers, which are then stacked to form the FNO.

$$v_{t_{n+1}}(\mathbf{X}) = \left(K_l \circ \sigma_l \circ K_{l-1} \circ \sigma_{l-1} \circ \cdots \circ \sigma_1 \circ K_0\right) v_{t_n} \tag{15}$$

The final output $v_{t_{n+1}}(\mathbf{X})$ from the FNO network is transformed using a projection layer to give the state $u_{t_{n+1}}$.

$$u_{t_{n+1}}(\mathbf{X}) = Q\left(v_{t_{n+1}}(\mathbf{X})\right) \tag{16}$$

4.1.2 U-Nets

U-Nets are another common baseline model for approximating PDE solutions, based on a composition of convolutional neural networks (CNNs) within a U-shaped architecture with an encoding and decoding section. In the most general form, a U-Net of depth i can be represented by the recursive formula [40]

$$U_i = \mathcal{D}_i(C_{i-1}(E_{i-1}(v_i, \psi_{E,i-1}), \psi_{D,i-1}), \psi_{D,i}), \qquad i = 1, 2, 3... \tag{17}$$

where v_i represents input state variables, U_i represents the output state variables, $\mathcal{D}_i$ a decoding (e.g. transposed/up convolution) CNN layer, E_i is the encoding (e.g. down-convolution) CNN layer, C represents a concatenation, and ψ are the linear weights. A non-linear function (e.g. rectified linear units or ReLU) is used within each decoder or encoder layer. In this simplified equation, the input state v_i is encoded with E and then concatenated with the output of the decoder, $\mathcal{D}$, at the corresponding skip location, which then proceeds up to the next layer in the network. Figure 2 shows an example of this architecture for a depth of 4, with additional embeddings including kinetic energy and potentially further derivatives as inputs.

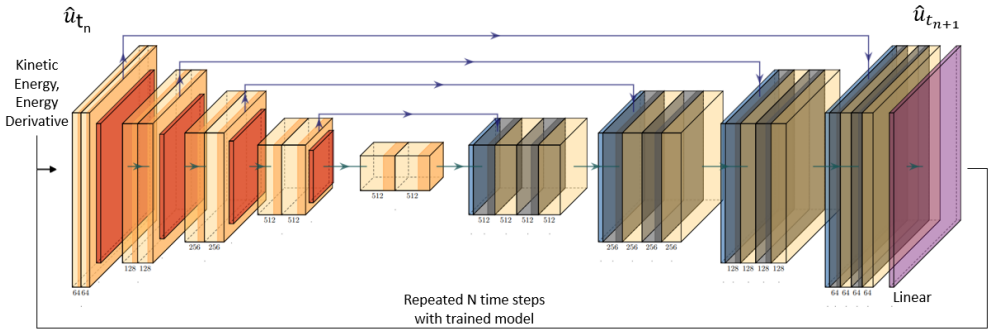

Figure 2: U-Net architecture visualized including encoder and decoder layers.

4.1.3 Scaling Displacements

During ML training, it is important to scale the displacements non-linearly because the magnitude of the central depression zone of the ball drop is over three orders of magnitude larger than the short-amplitude waves that propagate away from the depression zone. For stiff materials, small-amplitudes in displacements result in larger stress magnitudes, hence it is important to capture these effects. To mitigate this issue a non-linear piece-wise scaling was performed to transform the true results into a more suitable ML training space and then reverse the scaling at inference time.

$$u_{scaled,n} = \begin{cases} \text{sign}(u_n) \cdot \log(|u_n| + 1) & |u_n| > T \\ \frac{u_n}{T} & |u_n| \leq T \end{cases} \tag{18}$$

where T is a threshold providing sufficient balance between the range compression of the log transform and the expansion of the linear transform. Empirically, a constant of 1e-6 enables the ML to learn both the displacement at the impact zone and the waves that propagate away from the initial impact.

4.2 Physics-Guidance with Global Kinetic Energy Parameter

When computing the time-evolution of displacement fields, it is essential to incorporate information about the loading, which is derived from the ball-impact trajectory. To integrate this information into the machine learning strategy, a physics-based kinetic energy variable was embedded into the ML models. To forecast the global dynamics, a modified ResNet-18 model [41] was trained to predict the kinetic energy of the ball over the entire simulation time horizon. This predicted kinetic energy was then introduced into the spatial ML kernels as an embedding layer.

$$\mathcal{K}_{0:t_N} = f_{ResNet}(X) \tag{19}$$

The kinetic energy predictions, $\mathcal{K}$ and the partial time derivative of the kinetic energy where concatenated to the inputs of the displacement model, which significantly stabilized long time-horizon rollouts for both spatial ML kernels and resulted in appreciable improvement in the displacement predictions.

$$\hat{u}_{t_{n+1}} = f_{ML}(\mathcal{K}_{t_n}, \frac{\partial \mathcal{K}_{t_n}}{\partial t}, \hat{u}_{t_n}) \tag{20}$$

4.3 Time-Marching Strategies

Inspired by statistical time-series forecasting models [42], we explore different time evolution strategies and modifications to the spatial ML kernels. The basic auto-regressive strategy, where instead of a single time-step jump using a spatiotemporal model, the outputs at a certain time-steps are used as inputs for the next timestep are shown in Figure 3. By recursively applying this strategy, the entire time history is constructed.

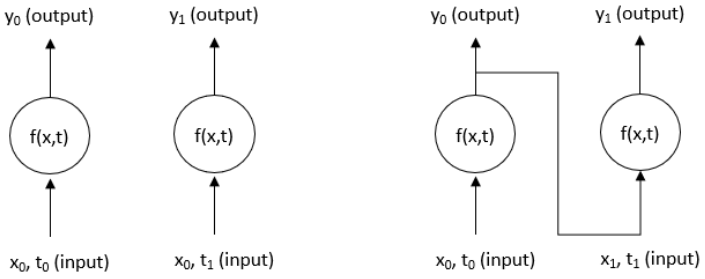

Figure 3: Different methods for physics surrogate modelling.

For numerical accuracy through the time, a balance has to be sought between capturing the finest temporal features based on the physics (such as local resonance time scales, or wave scattering phenomena) and maintaining stability during auto-regressive inference rollouts by avoiding accumulated error - which compounds with the total number of time-steps. Prior work in this area include the following time-marching strategies:

- **Temporal Bundling**: This method directly attempts to predict multiple future timesteps simultaneously $\{u_{t_n}, u_{t_{n+1}}, u_{t_{n+2}}\}$[43]. Similarly, a sequence of temporal inputs can be used (i.e. $u_{t_n} = f(u_{t_{n-1}}, u_{t_{n-2}}, ...)$) . The computational cost and memory requirements associated with this technique is prohibitive for training on large 3D datasets.
- **Pushforward Trick**: During training, the loss is modified such that a random distribution with variance, σ, perturbs the t sample. A possible method [43] for selection of the σ parameter is through the ML prediction noise. This method predicts multiple steps but only backpropagates temporal error on the last step. This methodology will be used in the later section, but proves insufficient to prevent temporal numerical instability for longer autoregressive rollouts.
- **Scheduled Sampling / Curriculum Learning**: In this method, the loss can randomly switch between truth and the ML prediction for training stability for the last temporal step in inputs. The exact distribution can be modified on a schedule. This will ensure that initially it is more likely to receive inputs from the true value, $u_{t_{n-1}}$, but as the training progresses, $\hat{u}_{t_{n-1}}$ will be chosen instead [31, 32]. This methodology results in training time complexity and requires heuristic tuning of schedules to ensure that the training is stable and convergent.
- **Temporal Multi-Resolution**: Our approach described in detail in 4.3.1, creates a series of models that have been trained at different resolutions or timesteps, Δt. The goal here is to create a coarse model that sacrifices accuracy for reduction in uncertainty propagation – then create a fine model that is temporally stabilized by the coarse model.

4.3.1 Temporal Multi-Resolution

The temporal multi-resolution method uses a series of trained neural networks with different time resolutions to reduce the accumulated error while capturing the physics sufficiently. Initial attempts to use a pure data-driven Markovian state prediction for the t_{n+1} change in displacements/strains using the solution at t_n, when used recursively to predict the entire transient series, exhibited a high degree of accumulated error. The auto-regressive models failed to accurately capture the global behavior when the ball bounced off the plate. However, significant uncertainty propagation caused the predictions to eventually diverge when using fine time-steps. A coarser temporal model with

$\Delta t_{coarse} > \alpha \cdot \Delta t_{fine}$, while unable to resolve physical phenomena in time, was found to mitigate the accumulated error empirically for $\alpha \geq 7$. In order to stabilize the finer resolution, while avoiding the damped predictions in the coarse time resolution, an interpolation between the coarser and finer predictions is proposed.

The proposed multi-resolution strategy offers several advantages including reduced error accumulation in long-time predictions and improved numerical stability; however it does have several limitations including an increased computational cost and heuristic selection of timesteps at each scale. Alternative approaches like curriculum learning and teacher forcing were explored to quantify the advantages of this approach but showed convergence issues at the scales considered here and hence not presented as a direct comparison in this study.

5 Results and discussion

Due to the lack of PDE benchmarks for scientific machine learning in impact simulations and elastodynamics, synthetic training data was created. This synthetic data generated through Moose were used for the results presented in this paper, and further information can be found in Appendix A.1. Based on an 80/20 split of the generated data, all statistics and evaluation were done on the test set. Training was done using the full temporal evolution period (0.5 seconds) for a given simulation. A brief review of the infrastructure and compute involved is given in Appendix A.2. On average, MOOSE simulations took approximately 5 hrs. For comparison, inference time of the same simulation is on the order of 1 second, demonstrating on the order of 10,000x speedup, with further details presented in Table 1.

5.1 Model Performance

Table 1 provides metrics for model assessment, and shows the performance of each model. These metrics are the maximum displacement error on an individual timestep basis, and a temporal stability parameter indicating the error across all the timesteps. For the purposes of understanding failure of material, the maximum displacement error is considered more important than the mean squared error which considers regions without any displacements.

Table 2 (in the Appendix) provides a performance/cost comparison for running FEM models and requirements of training. While ML models achieve approximately 10,000x speedup during inference (approx. 1 second vs 5 hours), this must be considered alongside the upfront costs of generating training data (approx. 11 hours on 156,000 CPU cores) and model training (approx. 17-30 hours on 56 GPUs). The ML approach becomes advantageous when running many similar simulations, as the initial overhead is amortized across multiple predictions. For applications requiring fewer simulations or significantly different geometries/materials, traditional FEM may be more practical over a machine learning approach.

Characterization of the temporal stability was formulated as the average accumulation of the error over the entire transient simulation.

$$\overline{\epsilon} = \frac{1}{M} \sum_{m=0}^{M} \left(\sum_{n=0}^{N} \frac{\epsilon_{m,n}^2}{N} \right)^{0.5} \tag{21}$$

where $\epsilon_{m,n} = max(u_{t_n}) - max(\hat{u}_{t_n})$, M is the total number of simulations in the test set, N is the total number of timesteps in a simulation.

In terms of the choice of timestep for time-marching, a coarse and fine set of values were used, where $\Delta t_{fine} = 0.01$ and $\Delta t_{coarse} = 0.07 = 7 \cdot t_{fine}$. This choice allowed a study of the effects of timestep size on numerical accuracy and stability of the method.

For the t_{fine} displacement models, the U-Net and FNO models exhibit an error of 1.2% and 4.9% respectively in terms of single step maximum displacement as shown in Table 1. The superior performance of U-Net in this particular case might be attributed to the local nature of the problem, where there is a large displacement surrounding the impact point but small displacements elsewhere, which is difficult to capture with a small set of Fourier modes. Each model architecture type greatly benefited from the use of cosine learning rate schedulers. Without any scheduling, each model type

learned the average simulation response, but often failed to predict the maximum values or the full range of the simulation results.

The reported error margins represent aggregate accumulated across all test cases. A more detailed error analysis on a case basis reveals that predictions are most accurate for impacts away from material interfaces and boundaries, while scenarios involving direct impacts near inclusion edges or corners show higher errors. This suggests that the ML models struggle most with capturing sharp discontinuities in material properties and geometric features. Future work should focus on improving accuracy in these edge cases through data-augmentation, enhanced network architectures or physics-informed constraints.

It should be noted that further hyperparameter optimization and heuristic tuning of modes might improve the performance of the FNO models. As expected from autoregressive models, the fine time resolution models are temporally unstable for longer time-horizons, and by the final time the accumulated error shows divergence for both the U-Net and FNO models. The coarse time resolution models show higher error on a single timestep level, since they do not resolve the physics sufficiently during the time-evolution, but they are temporally more stable, highlighting the need for the multi-resolution methodology.

While the scope of this paper is limited to deterministic ML predictions without diving into uncertainty quantification needed to understand reliability for industrial applications, future implementations should incorporate uncertainty quantification through ensemble methods and Bayesian neural networks with probabilistic outputs and include dropout-based uncertainty estimation during inference. Additionally, physics-based bounds on predicted quantities (e.g. material yield surface criteria) would allow engineers to make risk-informed decisions based on confidence levels in model predictions.

For an unseen ball and inclusion location case, the 3D fields of displacement components are obtained and qualitatively compared to the ground truth results. The displacement values at the location of ball-impact, with a through-thickness cross-section, are shown in Figures 4, 5 and 6 respectively. These results are obtained with the coarser time models, which are much more stable than the finer timestep models. These results empirically indicate that the U-Net model accrues marginally less error than the FNO model across the time marching, particularly evident in the y-component of displacement (Fig. 5. It is interesting to note that neither ML model was able to capture the strong discontinuity at the interface as seen in the x-component results (Fig. 4 towards the bottom left corner. Overall, these indicate that further improvements need to be done in terms of mitigating temporal error propagation to obtain agreement with ground truth, for which the pushforward method and multi-resolution models are explored.

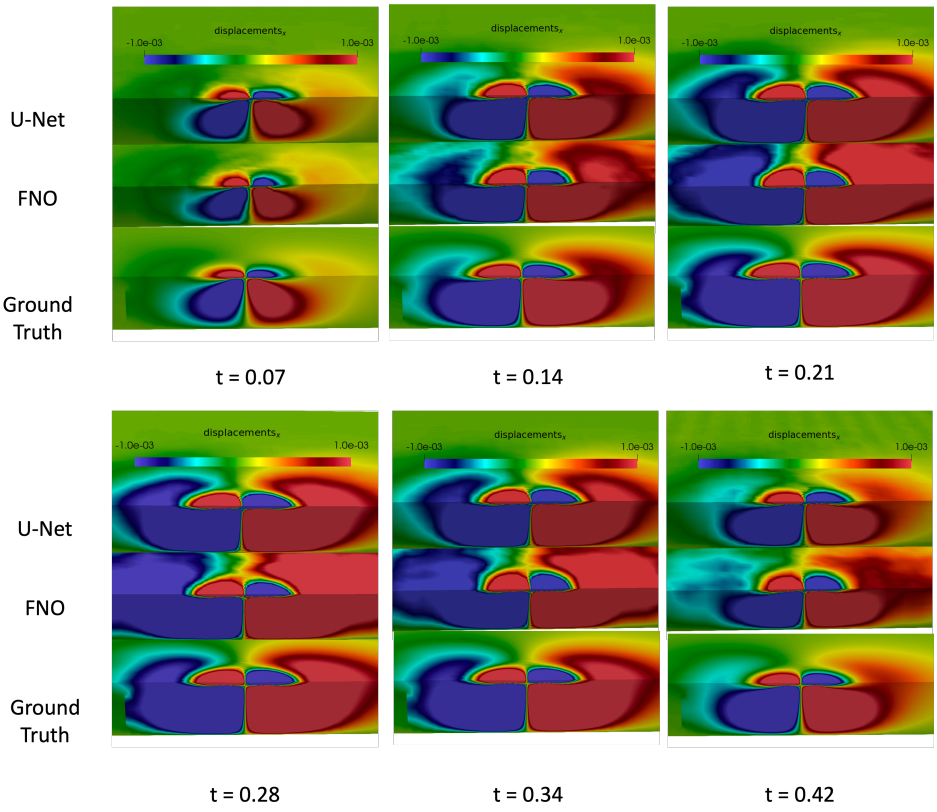

Figure 4: U-Net (t_{coarse}), FNO (t_{coarse}) and ground truth results for x-component of displacements (through thickness cross-section at impact point).

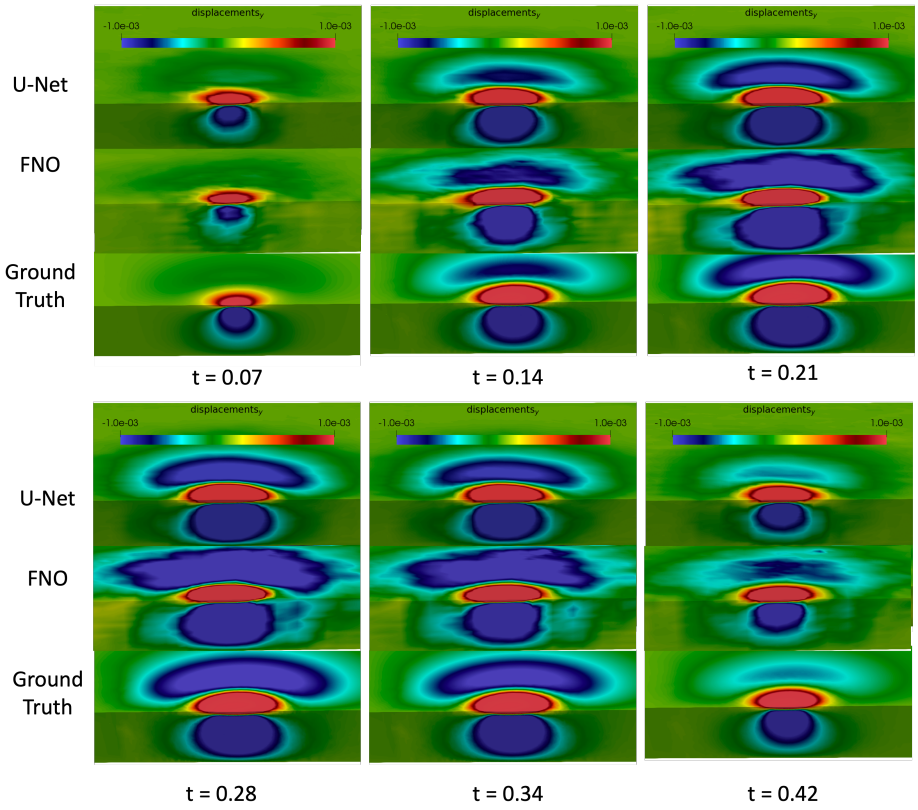

Figure 5: U-Net (t_{coarse}), FNO (t_{coarse}) and ground truth results for y-component of displacements (through thickness cross-section at impact point).

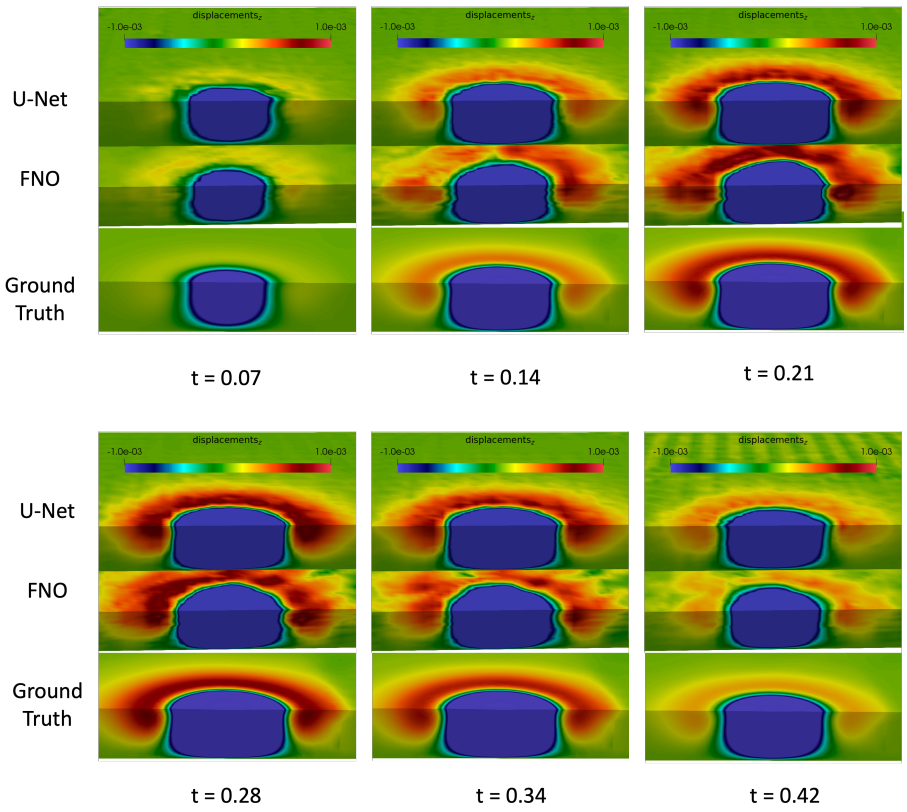

Figure 6: U-Net (t_{coarse}), FNO (t_{coarse}) and ground truth results for z-component of displacements (through thickness cross-section at impact point).

The pushforward method, also compared for the FNO and U-Net models, provides significant benefit to prevent the later timestep predictions from diverging, which is noted once again by the temporal stability error. However, the multi-resolution approach provides an improvement over using a single model with the push-forward method. With the multi-resolution setup, the coarse time models are hence used as a predictor-corrector setup for the t_{n+1} models. The resulting corrected models provide a balance between single-time error and longer-horizon stability as shown in Table 1. The figures for the final residuals of each model are provided in the Appendix A.4.

For quantitative analysis, the maximum displacement plots are obtained for each model. Due to the temporal instability seen in the FNO models (as indicated in Table 1), which cannot easily be resolved using the proposed multi-resolution strategy, the U-Net models are used to test the accuracy of the multi-resolution method. The maximum displacement, together with the corresponding kinetic energy of the ball showing the trajectory of impact, are shown for the corrected multi-resolution strategy in Fig. 7, showing a significantly improved performance over just using the fine temporal model in an autoregressive manner.

In an attempt to capture the temporal derivatives and understand how interpolation strategies compare to resolving finer-time scales numerically, a Hermitian spline of the displacement fields was computed and shown in Figure 8. These simple interpolations demonstrate that in the scenario the physics is well-behaved in between the $t_{n+\Delta t}$ predictions, temporal interpolation sufficiently provides missing physical information.

Network Arch	Max Disp. Error(%)	Temporal Stability(%)
U-Net | t_{fine}	1.2	2537.2
U-Net | t_{coarse}	7.7	21.5
U-Net$_{multi-res}$	–	8.7
FNO | t_{fine}	4.9	1422.1
FNO | t_{coarse}	7.4	27.5
FNO$_{multi-res}$	–	20.0
U-Net | t_{fine} pushforward	3.2	28.8
U-Net$_{multi-res}$ pushforward	–	10.0

Table 1: Comparison of model performance on test set.

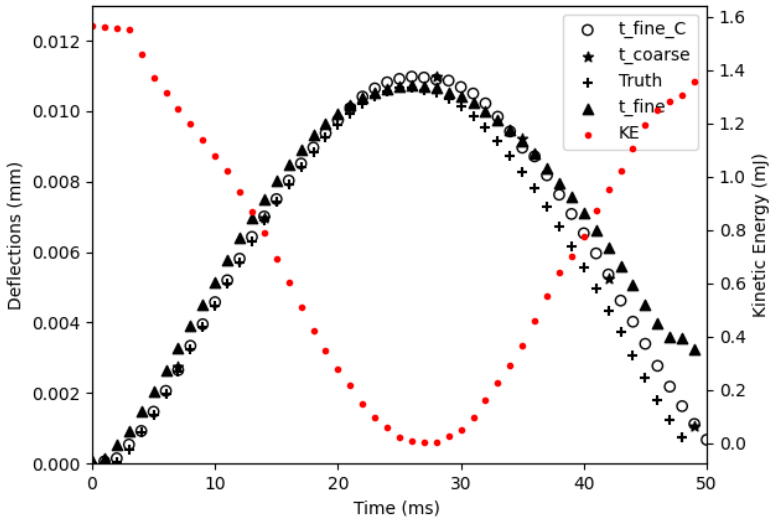

Figure 7: Corrected maximum of 3D predictions resulting in removal of propagation error for U-Net model

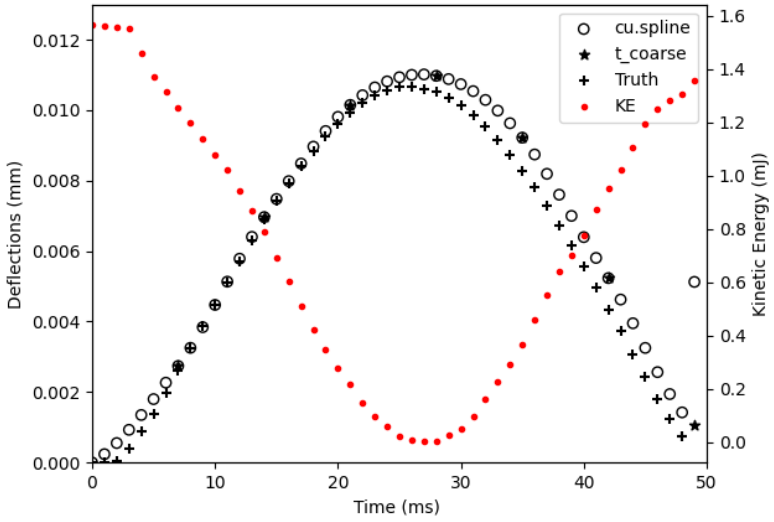

Figure 8: Hermite spline interpolation of the 3D field between coarse U-Net predictions

6 Conclusion

In this study, we have investigated the application of machine learning models to predict the transient structural response of an elastic medium subjected to dynamic impact loading. As an overview, the key findings from this study include:

1. Demonstrating that ML models can achieve 3.5-8% accuracy while providing significant computational speedup for ball-impact simulations

2. Devising a temporal multi-resolution strategy effectively mitigates error accumulation, but further investigation is required to understand how heuristic timestep selection can be tuned for different spatial kernels

3. As spatial ML kernels, U-Nets seem to show better temporal evolution stability than FNOs for long time-horizon predictions

The challenges associated with this problem, including discontinuities, and mesh-resolution dependent simulations were addressed through the generation of a large-scale converged synthetic dataset using an open-source finite element solver.

Using FNO and UNet architectures as spatial interpolation backbones, different time-marching strategies were explored in this work. The FNO architecture contained several key hyperparameters that affect model performance and stability: the number of Fourier layers, hidden features, and cutoff frequency modes. While systematic hyperparameter optimization through grid search or evolutionary algorithms could potentially improve FNO performance, the computational cost (approximately 17 hours per model on 56 GPUs) made extensive tuning impractical and hyperparameters were heuristically chosen. Future work will explore efficient hyperparameter optimization strategies balanced against computational constraints.

With standard autoregressive methodologies, the FNO method proved to be less numerically stable than the U-Net approach, although significant effort was not invested in tuning the hyperparameters of the FNO model. Various time-marching methods for data-driven models including temporal bundling and push-forward tricks were explored, yielding significant improvements over the classic autoregressive methodology. A multi-resolution strategy is proposed training individual neural networks to iteratively predict displacement at different temporal resolutions, leveraging a predictor-corrector framework to mitigate error propagation. The multi-resolution method demonstrated promising results, achieving less than 3.5-8% error in predicting maximum displacement compared to ground truth finite element simulations.

Through this work, we demonstrated the feasibility of data-driven transient physics predictions for structural dynamics problems. The utilization of synthetic data enabled the generalization of predictions to various shapes and designs not seen in the training set, highlighting the potential benefits of this approach for accelerating design optimization processes in industries such as consumer electronics.

While this work focused on primarily data-driven approaches, other than embedding a global kinetic energy parameter, incorporating further physics-informed constraints could improve model accuracy and stability. For elastodynamics, this includes enforcing compatibility conditions on the deformation gradient tensor and momentum conservation in the loss function. However, such constraints often introduce additional computational complexity and convergence challenges. Systematic exploration of physics-informed approaches is crucial, particularly investigating how to balance physical accuracy with computational efficiency.

Future work should also expand this study to incorporate more realistic industrial conditions including (a) material non-linearities such as plasticity and rate-dependent effects, (b) adaptive meshing for improved resolution in high-strain regions, (c) complex boundary conditions representative of device mounting, (d) multi-material interfaces with contact and debonding and (e) thermal effects and coupled thermomechanical behavior.

7 Acknowledgements

The authors are grateful to the following individuals: Shankar Ganapathysubramanian, Srinivas Tadepalli, Karthik Kumar and Barry Bolding. The authors would like to acknowledge the support of Amazon Web Services and Amazon Lab126 for providing guidance and computational resources.

References

[1] E Suhir and R Ghaffarian. Dynamic response of electronic materials to impact loading. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 97(6):699–717, 2017.

[2] Mohammad I Younis, Daniel Jordy, and James M Pitarresi. Computationally efficient approaches to characterize the dynamic response of microstructures under mechanical shock. *Journal of Microelectromechanical Systems*, 16(3):628–638, 2007.

[3] Mohammad A. Gharaibeh, Quang T. Su, and James M. Pitarresi. Analytical model for the transient analysis of electronic assemblies subjected to impact loading. *Microelectronics Reliability*, 91:112–119, 2018. ISSN 0026-2714. doi: https://doi.org/10.1016/j.microrel.2018.08.009. URL `https://www.sciencedirect.com/science/article/pii/S0026271418303329`.

[4] PNB Reis, JAM Ferreira, and NFS Rodrigues. Impact behaviour of panels for automotive applications. *Strain*, 47:79–86, 2011.

[5] Javid Bayandor, RS Thomson, ML Scott, MQ Nguyen, and DJ Elder. Investigation of impact and damage tolerance in advanced aerospace composite structures. *International journal of crashworthiness*, 8(3):297–306, 2003.

[6] Jingshi Meng, T Mattila, Abhijit Dasgupta, M Sillanpaa, Reino Jaakkola, K Andersson, and Esa Hussa. Testing and multi-scale modeling of drop and impact loading of complex mems microphone assemblies. In *2012 13th International Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems*, pages 1–8. IEEE, 2012.

[7] David E Lambert and C Allen Ross. Strain rate effects on dynamic fracture and strength. *International Journal of Impact Engineering*, 24(10):985–998, 2000.

[8] Tod A Laursen. *Computational contact and impact mechanics: fundamentals of modeling interfacial phenomena in nonlinear finite element analysis*. Springer Science & Business Media, 2013.

[9] Peyman Rafiee, Golta Khatibi, and Michael Zehetbauer. A review of the most important failure, reliability and nonlinearity aspects in the development of microelectromechanical systems (mems). *Microelectronics International*, 34(1):9–21, 2017.

[10] VBC Tan, MX Tong, Kian Meng Lim, and Chwee Teck Lim. Finite element modeling of electronic packages subjected to drop impact. *IEEE transactions on components and packaging technologies*, 28(3):555–560, 2005.

[11] Kamarajan Balakrishnan, Arvind Sharma, and Rajib Ali. Comparison of explicit and implicit finite element methods and its effectiveness for drop test of electronic control unit. *Procedia engineering*, 173:424–431, 2017.

[12] Delfim Soares Jr. A model/solution-adaptive explicit-implicit time-marching technique for wave propagation analysis. *International Journal for Numerical Methods in Engineering*, 119(7): 590–617, 2019.

[13] Anthony Gravouil, Thomas Elguedj, and Hubert Maigre. An explicit dynamics extended finite element method. part 2: Element-by-element stable-explicit/explicit dynamic scheme. *Computer Methods in Applied Mechanics and Engineering*, 198(30-32):2318–2328, 2009.

[14] Thomas Grätsch and Klaus-Jürgen Bathe. A posteriori error estimation techniques in practical finite element analysis. *Computers & structures*, 83(4-5):235–265, 2005.

[15] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[17] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[18] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[19] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[21] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3), May 2024. doi: 10.1145/3648506. URL `https://doi.org/10.1145/3648506`.

[22] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[23] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, 147(8):04021043, 2021.

[25] Majid Rasht-Behesht, Christian Huber, Khemraj Shukla, and George Em Karniadakis. Physics-informed neural networks (pinns) for wave propagation and full waveform inversions. *Journal of Geophysical Research: Solid Earth*, 127(5):e2021JB023120, 2022.

[26] Fanny Lehmann, Filippo Gatti, Michaël Bertin, and Didier Clouteau. 3d elastic wave propagation with a factorized fourier neural operator (f-fno). *Computer Methods in Applied Mechanics and Engineering*, 420:116718, 2024.

[27] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[28] Phillip Lippe, Bastiaan S Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. Modeling accurate long rollouts with temporal neural pde solvers. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[29] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pderefiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[31] Lalit Ghule, Rishikesh Ranade, and Jay Pathak. Nlp inspired training mechanics for modeling transient dynamics. *arXiv preprint arXiv:2211.02716*, 2022.

[32] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[33] Cody J Permann, Derek R Gaston, David Andrš, Robert W Carlsen, Fande Kong, Alexander D Lindsay, Jason M Miller, John W Peterson, Andrew E Slaughter, Roy H Stogner, et al. Moose: Enabling massively parallel multiphysics simulation. *SoftwareX*, 11:100430, 2020.

[34] Alexander D Lindsay, Derek R Gaston, Cody J Permann, Jason M Miller, David Andrš, Andrew E Slaughter, Fande Kong, Joshua Hansel, Robert W Carlsen, Casey Icenhour, et al. 2.0-moose: Enabling massively parallel multiphysics simulation. *SoftwareX*, 20:101202, 2022.

[35] Guillaume Giudicelli, Alexander Lindsay, Logan Harbour, Casey Icenhour, Mengnan Li, Joshua E Hansel, Peter German, Patrick Behne, Oana Marin, Roy H Stogner, et al. 3.0-moose: Enabling massively parallel multiphysics simulations. *SoftwareX*, 26:101690, 2024.

[36] Stanislas Zaremba. Sur une forme perfectionee dela theorie de la relaxation. *Bull. Int. Acad. Sci. Cracovie*, 1903.

[37] Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. *Nonlinear finite elements for continua and structures*. John wiley & sons, 2014.

[38] MM Rashid. Incremental kinematics for finite element applications. *International Journal for Numerical Methods in Engineering*, 36(23):3937–3956, 1993.

[39] Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *arXiv preprint arXiv:2306.10619*, 2023.

[40] Christopher Williams, Fabian Falck, George Deligiannidis, Chris C Holmes, Arnaud Doucet, and Saifuddin Syed. A unified framework for u-net design and analysis. *Advances in Neural Information Processing Systems*, 36, 2024.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] Paul Newbold. Arima model building and the time series analysis approach to forecasting. *Journal of forecasting*, 2(1):23–35, 1983.

[43] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[44] S Rossi, N Abboud, and G Scovazzi. Implicit finite incompressible elastodynamics with linear finite elements: A stabilized method in rate form. *Computer Methods in Applied Mechanics and Engineering*, 311:208–249, 2016.

[45] Zhiwen Chen and Baorong Zhong. Tfinterpy: A high-performance spatial interpolation python package. *SoftwareX*, 20:101229, 2022.

[46] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. volume 79, pages 1309–1331, 2009.

[47] Sandia National Laboratories. Seacas: A computer aided engineering software for analyzing structural mechanics and thermal analysis finite element models. 2023. GitHub repository: https://github.com/sandialabs/seacas.

A Synthetic data generation

A.1 MOOSE simulations

The numerical simulations are implemented using the open-source MOOSE library [33–35], using an implicit time marching methodology to avoid accumulated error and numerical instabilities [44]. The setup of numerical experiments is shown in Figure 1, with a solid plate with an embedded laminate inclusion impacted by a spherical ball.

The ball, plate, and inclusion exhibit material properties representing acrylic, polycarbonate, and glass. The ball is dropped from a height of $\mathcal{L} = 16$ cm above the plate. A total of 6500 randomized designs were generated by varying 8 variables: each design exhibited a different ball location (X_b, Y_b), inclusion centroid location (X_i, Y_i, Z_i), and inclusion dimensions (L_i, D_i, H_i). For the FEM calculations, to synthesize training data, a total of approximately 50,000 CPU cores was utilized across 2000 compute nodes (using either AMD EPYC 9R14 or Intel Xeon 8488C processors) creating 30TB of data in approximately 11 hours. A series of mesh sensitivity studies were performed to ensure the ground truth simulations were converged. Mesh contact was iteratively tested and examined to prevent mesh nodal interpenetration. The solver used a fully implicit Newton-Krylov solver with an adaptive time-step for a total simulation time of 0.5 seconds. Each simulation used identical initial conditions for the ball mass and drop height. The ball contact was modelled with 0.2 coefficient of friction while the inclusion had a fixed boundary condition to tie it with the surrounding material.

Once the numerical results were obtained, the unstructured mesh solutions were resampled onto a uniform structured grid through an inverse distance weighting algorithm through the TFInterpy Python library [45] in order to train the U-Net and FNO machine learning strategies.

Mesh generation was achieved by using the python API for the gmsh [46] meshing tool. A python script was created that enabled procedural generation of the ball, plate, and inclusion. The mesh used tetrahedral elements with a maximum element size of 3mm. A virtual box was centered at the point of ball impact with a size of 3mm X 3mm X height of outer prism plus 3mm. This virtual box was then used to further refine the mesh in which the maximum element size was reduced to 0.2mm, which includes the bottom portion of the ball. The gmsh tool then gradually increased the mesh size over a range of 50mm to the global max element size of 3mm. Mesh sensitivity studies concluded this meshing scheme to be sufficient for convergence for the variety of possible designs that could be generated.

The contact locations between the upper surface and ball or inclusion and outer prism used a mesh penalty approach. A sensitivity study was performed on these penalties, by increasing them until no penetration was observed for a variety of generated meshes.

The total of 6500 simulations is a somewhat arbitrary number based on $\sim 3^8$ data points needed to understand interactions with 8 variables. Out of the 6500 cases, an 80/20 split was used for training and testing of all neural networks. No hyperparameter tuning was used with the test set, which was only used for final evaluation. The SEACAS [47] tool was used to extract all MOOSE Exodus mesh output, which enabled quick conversion for ML preprocessing. The final distributions of the training and test set can be viewed in Figure 9. Visualization was performed using ParaView with custom Python macros for extracting temporal sequence slices.

A.2 Data generation / training costs

Using on-demand pricing with AWS cloud infrastructure, the compute requirements for this study are provided in Table 2. While a thorough investigation of run times and costs over all types of compute instances is interesting future work, Table 2 serves as an estimate. It was observed that G5 instances (A10G GPUs) outperformed the P3 instances (V100 GPUs). The higher CPU clock speeds and more CPUs available for loading and unloading data, is an important bottleneck compared to the end-to-end inference time of the physics surrogate models.

A.3 Preprocessing of data

The workflow to train the model at a large scale commences with data preprocessing, which includes scaling and voxelization using inverse distance weighting. For computational efficiency, an adaptive timestep approach was employed within the MOOSE framework. This introduced the challenge of

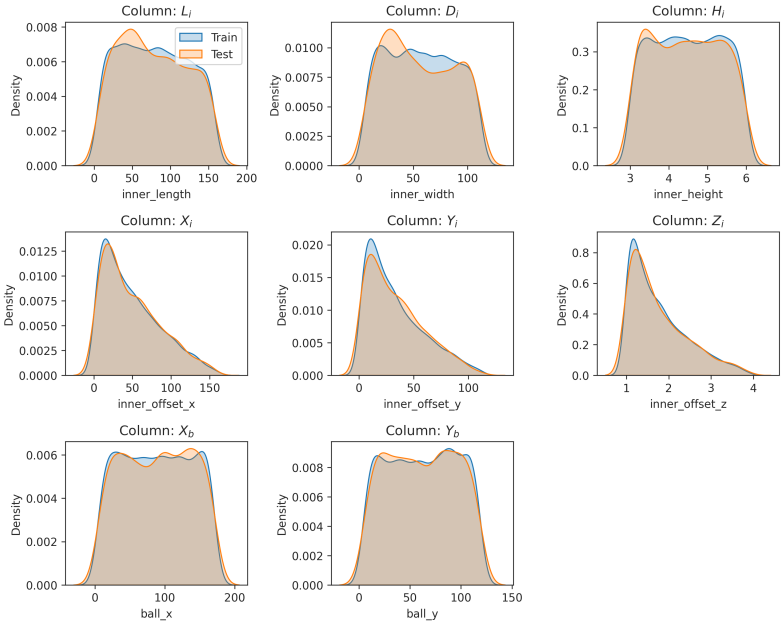

Figure 9: Distribution of the training and test sets used for the ML training/testing.

Task	N cpu	N GPU	Time (hrs)	Cost ($)
MOOSE simulations	156e3	–	11	26,900
Storage (30TB)	–	–	–	700
GPU (G5)	1344	56	17	1,900
GPU (P3)	672	56	30	5,100

Table 2: Comparison of infrastructure costs

inconsistent timestep counts across different simulations. A preprocessing interpolation step was used to create a uniform set of timesteps for training. By adopting this approach, the network learns the incremental changes in the quantity of interest ($\partial\phi$) instead of its time derivative ($\partial\phi/\partial t$), reducing both the number of input features and potential variation in the results.

The geometry does not require high resolution voxelization and thus 128 X 128 X 16 was found to be sufficient. This resolution results in an effective voxel size of Δx of 1.38mm, Δy of 0.97mm, and Δz of 0.5mm.

A.4 Figures of comparison

Further analysis of results to support the findings in the main text are presented here. Figures 10, 11, 12 and 13 show the deviation of maximum displacement in the predictions against the ground truth for the unseen test set. Large off-diagonal deviations typically result from corner-cases where the impact occurs on or close to the edge of inclusion.

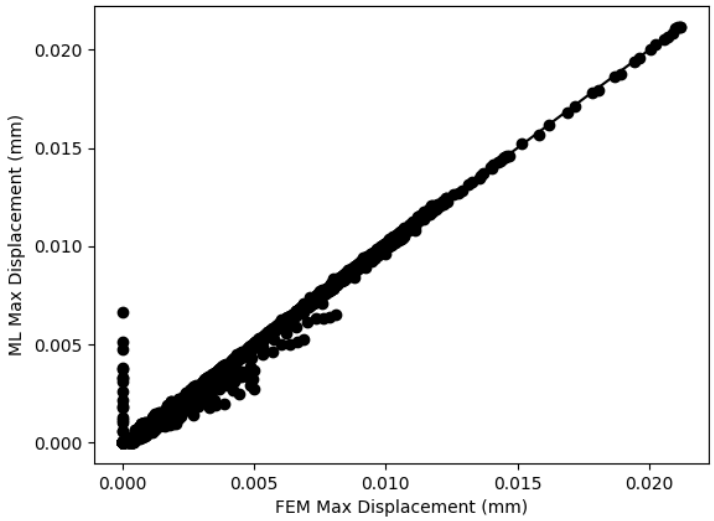

Figure 10: Test set predictions for the FNO t_{fine} model for the maximum displacement.

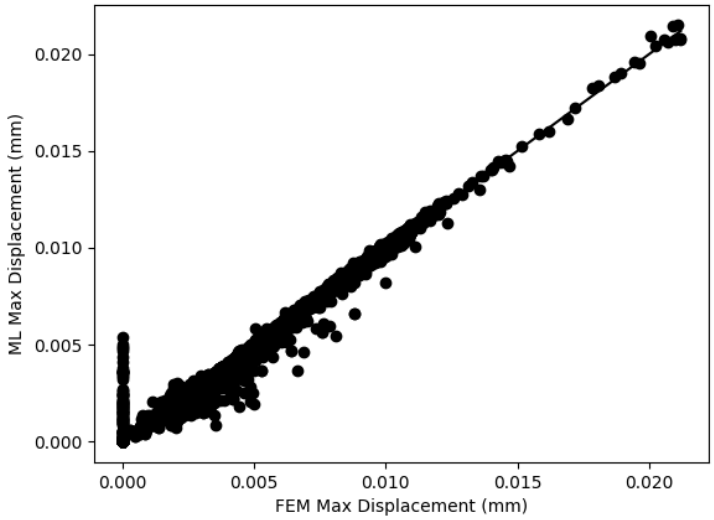

Figure 11: Test set predictions for the FNO t_{coarse} model for the maximum displacement.

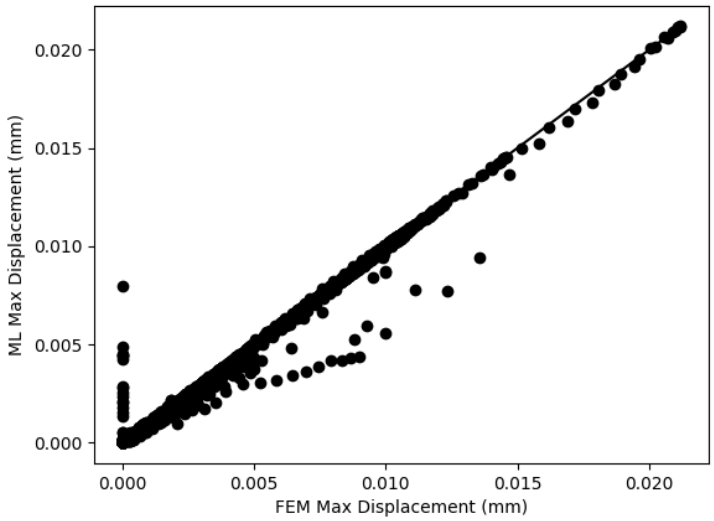

Figure 12: Test set predictions for the U-Net t_{fine} step model for the maximum displacement.

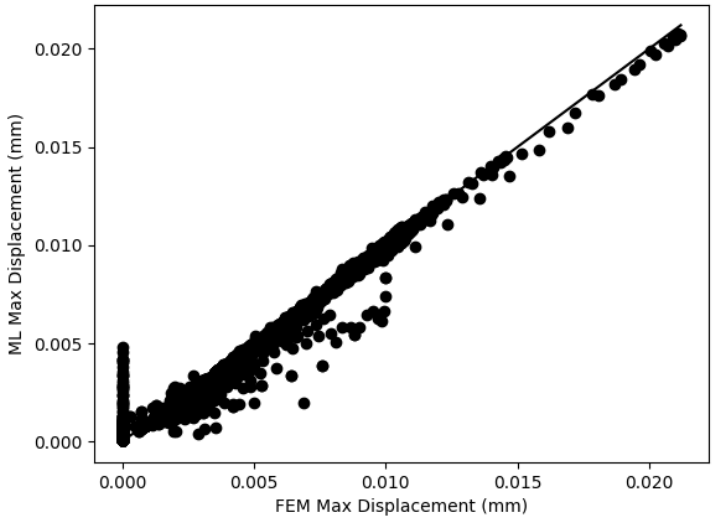

Figure 13: Test set predictions for the U-Net t_{coarse} step model for the maximum displacement.